UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Mateus Martelini Souza

Processamento e Classificação de Imagens Multiespectrais de Lâminas Histológicas coradas com H&E

São Carlos

2021

Mateus Martelini Souza

Processamento e Classificação de Imagens Multiespectrais de Lâminas Histológicas coradas com H&E

Monografia apresentada ao Curso de Engenharia Elétrica com Ênfase em Eletrônica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Sebastião Pratavieira

São Carlos

2021

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Martelini Souza, Mateus
M376p Processamento e classificação de imagens multiespectrais de lâminas histológicas coradas com H&E / Mateus Martelini Souza; orientador Sebastião Pratavieira. São Carlos, 2021.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2021.

1. Imagem multiespectral. 2. Lâminas histológicas. 3. Histopatologia. 4. Aprendizado de Máquina. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Nome: Mateus Martelini Souza

Título: "Processamento e Classificação de Imagens Multiespectrais de Lâminas Histológicas coradas com H&E"

Trabalho de Conclusão de Curso defendido e aprovado em 23 / 07 / 2021,

com NOTA 9,6 (nove, seis), pela Comissão Julgadora:

Prof. Dr. Sebastião Pratavieira - Orientador - FMC/IFSC/USP

Prof. Dr. Marlon Rodrigues Garcia - SEL/EESC/USP

Prof. Associado Daniel Varela Magalhães - SEM/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

Dedico este trabalho a todos os que me ajudaram ao longo desta caminhada.

AGRADECIMENTOS

Agradeço primeiramente a Deus, que me deu forças e capacidade para que eu pudesse realizar essa graduação e chegar até o final.

Agradeço aos meus pais e à minha irmã, que sempre estiveram ao meu lado, me incentivaram e se esforçaram para que eu pudesse realizar esse curso.

Agradeço à minha noiva por ter me apoiado em tudo e por ter estado sempre ao meu lado.

Agradeço à Universidade de São Paulo pelos recursos oferecidos, que foram indispensáveis para a minha formação.

Agradeço aos professores que dedicaram o tempo a ensinar a mim e aos demais alunos.

Em especial, agradeço ao professor Marlon, que sempre demonstrou um cuidado especial pelos alunos, buscando sempre nos incentivar a aprender mais.

Também agradeço ao professor Sebastião, que se mostrou disponível para ajudar quando necessário.

Agradeço a todos os meus colegas da graduação, que tornaram esse caminho muito mais divertido e leve.

Agradeço aos colegas de pesquisa Felipe, Enzo e Bruno, que contribuíram para esse trabalho.

Por fim, agradeço a todos aqueles que de alguma forma contribuíram para a minha formação, seja me apoiando na universidade ou fora dela.

"A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda pensou sobre aquilo que todo mundo vê."

Arthur Schopenhauer

RESUMO

SOUZA, M. M. **Processamento e Classificação de Imagens Multiespectrais de Lâminas Histológicas coradas com H&E**. 2021. 67p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2021.

As imagens multiespectrais são compostas de diversos canais, que representam comprimentos de onda diferentes e possuem informações espectrais que podem ser utilizadas na classificação de tecidos de lâminas histológicas por algoritmos de Aprendizado de Máquina. Atualmente o padrão ouro para o diagnóstico de histopatologias é a análise da lâmina histológica pelo patologista. O objetivo desse trabalho foi usar algoritmos de Processamento de Imagem e Inteligência Artificial para auxiliar esses profissionais na identificação dos tecidos e de regiões de tumor. Para isso, uma instrumentação composta de um microscópio, um filtro óptico e uma câmera monocromática foi controlada pelo computador através de um algoritmo em LabVIEW para realizar a aquisição das imagens multiespectrais de pele de camundongo coradas com H&E. O algoritmo de processamento e classificação foi desenvolvido em linguagem de programação *Python*. Os modelos de Aprendizado de Máquina utilizados foram o *K-Nearest Neighbors*, o *Support Vector Machine* (SVM) com núcleo linear, o SVM com núcleo RBF e o *Random Forest*. Os filtros passa-baixa e da mediana foram utilizados como pré e pós-processamento, respectivamente. A taxa de acerto para os modelos *Random Forest* e *K-Nearest Neighbors* foi de 0,93 com a aplicação de ambos os filtros. Conclui-se que foi possível desenvolver um algoritmo classificador de lâminas histológicas com boa taxa de acerto para auxiliar os histopatologistas na identificação dos tecidos e das regiões de tumor.

Palavras-chave: Imagem multiespectral. Aprendizado de Máquina. Lâminas histológicas. Histopatologia.

ABSTRACT

SOUZA, M. M. **Processing and Classification of Multispectral Images of H&E-Stained Histological Slides**. 2021. 67p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2021.

Multispectral images are composed of several channels, which represent different wavelengths and have spectral information that can be used in the classification of tissues of histological slides by Machine Learning algorithms. Currently the gold standard for the diagnosis of histologies is the analysis of the histological slide by the pathologist. The objective of this work was to use Image Processing and Artificial Intelligence algorithms to assist these professionals in the identification of tissues and tumor regions. For this, an instrumentation composed of a microscope, an optical filter and a monochrome camera was controlled by the computer through a LabVIEW algorithm to perform the acquisition of multispectral images of H&E-stained mouse skin. The processing and classification algorithm was developed in Python. The Machine Learning models used were K-Nearest Neighbors, Support Vector Machine (SVM) with linear kernel, SVM with RBF kernel, and Random Forest. The low-pass and median filters were used as pre and post-processing, respectively. The accuracy for the Random Forest and K-Nearest Neighbors models was 0.93 with the application of both filters. It was concluded that it was possible to develop a classifier algorithm of histological slides with good accuracy to aid histopathologists in the identification of tissues and tumor regions.

Keywords: Hyperspectral image. Machine Learning. Histological slides. Histopathology.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

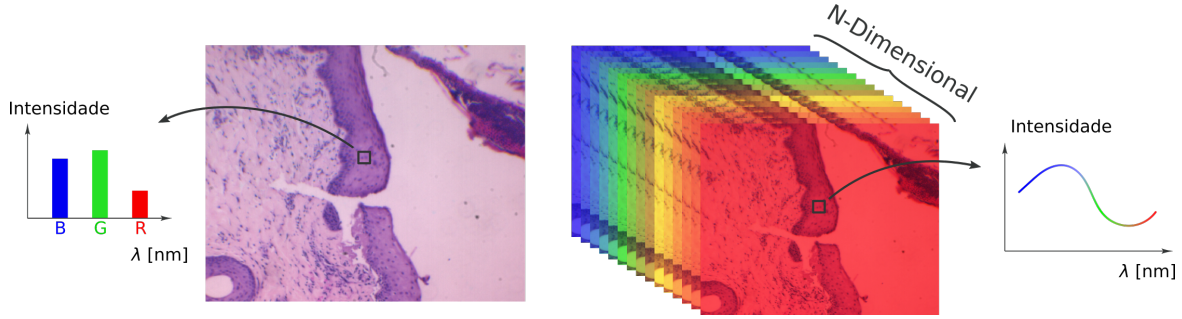
H&E	Hematoxilina-Eosina
KNN	K-Nearest Neighbors
LCTF	Liquid Crystal Tunable Filter
RBF	Radial Basis Function
RF	Random Forest
RGB	Red Green Blue
SVM	Support Vector Machine
USB	Universal Serial Bus

SUMÁRIO

1 INTRODUÇÃO

As imagens multiespectrais são imagens formadas por múltiplos canais. Cada canal é constituído de uma imagem monocromática capturada em um comprimento de onda específico proveniente do objeto imageado. Quando se utiliza diversos canais, obtém-se informações espaciais e espectrais do objeto imageado. Assim, cada pixel da imagem apresenta N valores, sendo N o número de canais. Essa curva de valores representa a assinatura espectral da região captada por um determinado pixel. A imagem com canais RGB puros é um caso particular de imagem multiespectral, composta por apenas três canais, os quais correspondem aos comprimentos de onda das cores vermelha, verde e azul (ORTEGA *et al.*, 2020). A figura 1 ilustra esse conceito.

Figura 1 – Diagrama de uma imagem multiespectral e comparação com a imagem RGB



Fonte: Do próprio autor

Cada molécula ou substância possui uma curva de absorção, ou seja, absorve alguns comprimentos de onda e permite a passagem ou reflexão de outros. Essa curva é específica para cada molécula, logo, o imageamento multiespectral permite detectar certas substâncias com mais precisão devido à maior quantidade de informação espectral (MANSFIELD, 2014).

Uma das técnicas de obtenção de uma imagem multiespectral consiste em utilizar um filtro ajustável eletronicamente que permite a passagem de apenas um comprimento de onda por vez. Esse filtro é chamado de LCTF (do inglês *Liquid Crystal Tunable Filter*) e pode ser controlado pelo computador via USB. Assim, é possível obter diversas imagens monocromáticas em sequência, cada uma representando a transmissão ou reflexão do objeto imageado para um determinado comprimento de onda.

As lâminas histológicas são lâminas de vidro que contém uma fatia de um determinado tecido que pode ser analisado via miscroscópio, por exemplo. Esses tecidos podem passar por uma etapa de coloração com o objetivo de aumentar o contraste entre as diversas estruturas biológicas presentes e, assim, facilitar a visualização. Um exemplo

de coloração muito comum é com as substâncias Hematoxilina e Eosina (H&E).

Atualmente, a análise de lâminas histológicas por patologistas é o padrão ouro para o diagnóstico de patologias em tecidos. A análise também é realizada para acompanhar o progresso dos tratamentos realizados em pacientes com tumores. No entanto, essa análise é subjetiva e depende do conhecimento do profissional. Considerando que durante a evolução de uma patologia ou tratamento o tecido sofre alteração das suas propriedades ópticas e espectrais, é possível utilizar o imageamento multiespectral como uma ferramenta auxiliar no diagnóstico de patologias (LU; FEI, 2014).

O Processamento Digital de Imagens consiste de algoritmos que realizam transformações nas informações de uma imagem de forma a obter uma outra imagem ou informações extraídas dela (GONZALEZ; WOODS; EDDINS, 2007). Alguns desses algoritmos, como os filtros gaussianos, podem ser utilizados como pré-processamento, que é uma etapa inicial que prepara a imagem para o processamento principal.

O algoritmo principal pode ser um modelo de Aprendizado de Máquina, que utiliza algumas imagens como conjunto de dados de treinamento e, a partir delas, ajusta parâmetros internos do modelo para realizar um determinado objetivo, que pode ser classificação, segmentação, identificação de objetos e regiões, entre outros. Após o treinamento, o modelo é capaz de realizar a tarefa para outras imagens. Os modelos de Aprendizado de Máquinas são mais adequados para processar uma grande quantidade de dados, pois isso permite que eles ajustem com mais precisão os parâmetros internos e atinjam melhores resultados para diferentes tipos de imagens.

No contexto das lâminas histológicas, o desenvolvimento de algoritmos que segmentam e classificam os tecidos biológicos e fornecem informações quantitativas, como a área da região danificada, pode ajudar na análise e no diagnóstico de patologias. Além disso, as imagens multiespectrais possuem muita informação, o que dificulta a performance de algoritmos de processamento mais simples. Portanto, os algoritmos de Aprendizado de Máquina são adequados para realizar esse tipo de atividade.

1.1 Objetivos

1.1.1 Objetivo Geral

Este projeto tem por objetivo desenvolver um algoritmo de processamento de imagens e aprendizado de máquina para realizar a classificação de imagens multiespectrais de lâminas histológicas a fim de ajudar os patologistas nos diagnósticos de patologias e no acompanhamento dos tratamentos.

1.1.2 Objetivos Específicos

- Fazer a aquisição das imagens multiespectrais que serão utilizadas no treinamento do modelo;
- Desenvolver um algoritmo que realiza o pré-processamento das imagens, se necessário;
- Desenvolver a etapa de treinamento do modelo de aprendizado de máquina;
- Desenvolver a etapa de pós-processamento para a visualização dos resultados.

1.2 Organização do trabalho

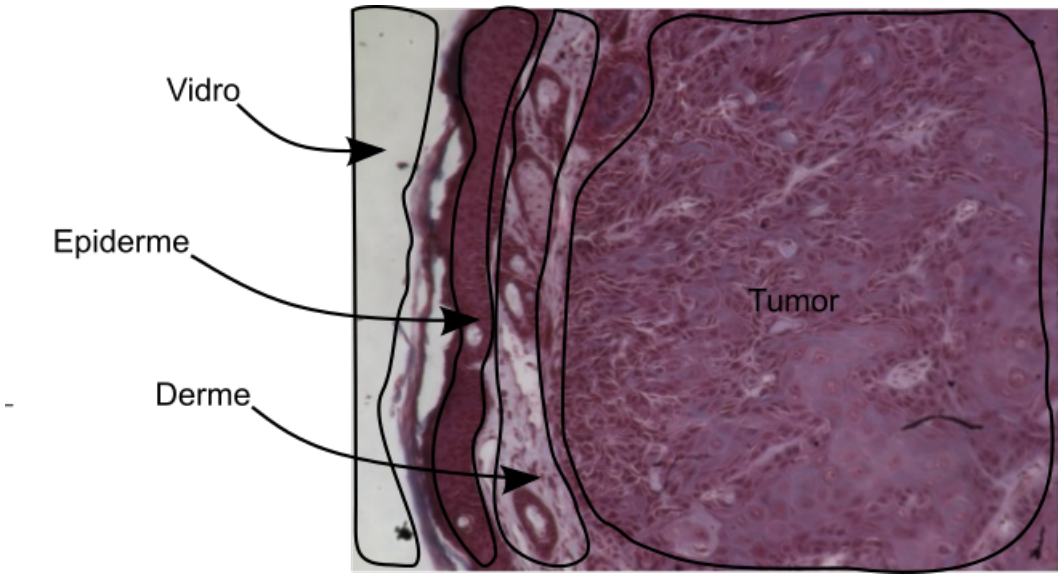
O capítulo 2 desse trabalho apresenta alguns conceitos fundamentais para o entendimento do algoritmo desenvolvido. O capítulo 3 descreve como as imagens foram adquiridas e como o modelo foi treinado. O capítulo 4 apresenta os resultados desse trabalho, que são as imagens adquiridas e as suas respectivas classificações. Por fim, o capítulo 5 discute as conclusões obtidas após a realização desse projeto.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Tecido biológico

Os tecidos biológicos utilizados para a classificação nesse projeto foram os que estão presentes na pele de camundongo. Os principais tecidos desse órgão são a epiderme e a derme. Outras estruturas que podem eventualmente aparecer são o colágeno, a queratina, os vasos sanguíneos, as hemácias, o extrato córneo, entre outros. Além disso, o algoritmo de classificação foi treinado para identificar regiões de tumor e regiões da lâmina em que não há nenhuma célula, chamada de região do vidro. A figura 2 mostra a foto obtida no microscópio de uma região de uma lâmina com algumas regiões da pele de camundongo nomeadas.

Figura 2 – Lâmina de pele de camundongo corada e identificação das regiões do tumor, da derme, da epiderme e do vidro



Fonte: Do próprio autor

2.2 Informação espectral

Cada molécula possui um espectro de emissão e de absorção característicos, que correspondem aos comprimentos de onda da radiação eletromagnética que são emitidos ou absorvidos por ela. Dessa forma, ao analisar o espectro de emissão de uma certa molécula, é possível identificá-la. Os corantes utilizados nas lâminas histológicas também possuem as suas respectivas curvas de absorção.

Cada tecido biológico é composto de diversas moléculas e seu espectro de absorção é formado por uma composição dos espectros das moléculas presentes naquele tecido.

Assim, ao iluminar o tecido com uma luz que possui componentes em todo o espectro (luz branca) e observar o espectro da luz transmitida, é possível identificar o tipo de tecido em estudo. As imagens multiespectrais possuem informações espectrais que podem ser utilizadas para identificar cada tecido com mais exatidão (FARKAS; BECKER, 2001). Esse foi o princípio utilizado nesse projeto.

2.3 Processamento digital de imagens

O Processamento Digital de Imagens compreende qualquer algoritmo digital que utiliza uma imagem como entrada e realiza alguma transformação de forma a obter outra imagem ou informações extraídas dela (GONZALEZ; WOODS; EDDINS, 2007). Uma das técnicas fundamentais de processamento de imagens são os filtros. Existem diversos tipos de filtros, que podem ser aplicados no domínio do espaço ou da frequência, que para imagens geralmente corresponde à variação da intensidade dos *pixels* no espaço.

Nesse projeto foi aplicado como pré-processamento um filtro passa-baixa no domínio do espaço, que foi constituído por um filtro da média. De forma simplificada, cada um dos pixels tem seu valor substituído pela média dos valores dos pixels vizinhos. Uma máscara de tamanho m x n é definida para determinar quantos e quais pixels vizinhos são usados na operação (GONZALEZ; WOODS; EDDINS, 2007). O resultado da aplicação desse tipo de filtro é uma suavização da imagem, que se torna visualmente mais embaçada.

As imagens utilizadas no projeto apresentam algumas irregularidades, como células, núcleos e outras estruturas pequenas. Porém, a aplicação visa classificar regiões maiores e mais uniformes. Portanto, o uso de um filtro passa-baixa poderia diminuir a interferência dessas estruturas e, por isso, esse tipo de filtro será testado.

Nesse trabalho, foi utilizado como técnica de pós-processamento o filtro da mediana, aplicado no domínio do espaço. De forma semelhante ao do filtro passa-baixa, cada pixel é substituído pela mediana dos pixels vizinhos. Também existe um máscara que determina quais são os pixels vizinhos a serem utilizados. Para o tipo de imagem desse trabalho, o resultado é semelhante ao do filtro passa-baixa. No entanto, os pixels da imagem classificada possuem um conjunto discreto de valores possíveis, que correspondem aos tecidos classificados e, por isso, é mais adequado utilizar a mediana ao invés da média, visto que a última pode gerar valores não inteiros.

2.4 Modelos de Aprendizado de Máquina

O Aprendizado de Máquina é a programação de computadores para que eles aprendam com os dados (GERON, 2017). Os modelos de Aprendizado de Máquina são constituídos de algoritmos computacionais que analizam um conjunto de dados com o objetivo de extrair alguma informação deles. Nesse projeto, buscou-se encontrar um modelo

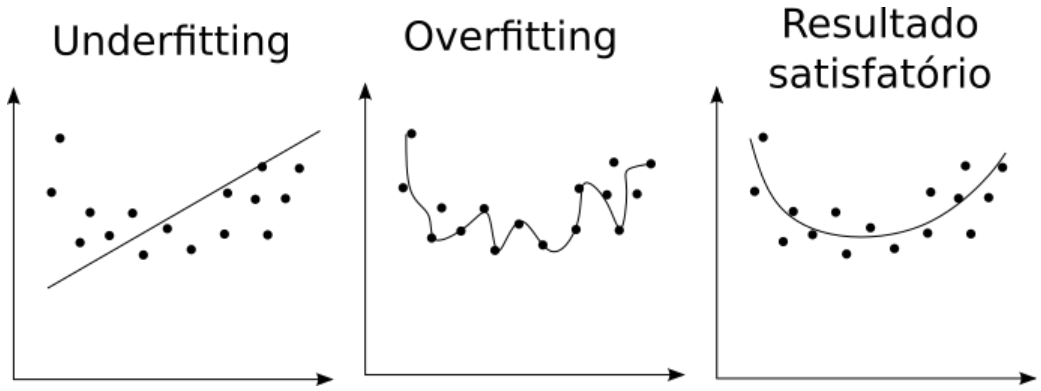
que aprendesse a classificar cada região da imagem de entrada como um determinado tecido da pele.

Existem dois tipos de algoritmos de Aprendizado de Máquina: o supervisionado e o não-supervisionado. O primeiro tem como entrada os dados a serem analisados e os dados que são desejados na saída do algoritmo. O segundo tipo de algoritmo necessita apenas dos dados de entrada e o algoritmo atua de forma a reconhecer padrões ou agrupar esses dados.

O processo de treinamento de um modelo de Aprendizado de Máquina consiste em fornecer os dados separados previamente e permitir que o algoritmo ajuste seus parâmetros internos de forma a realizar determinada atividade. Uma dessas atividades é a classificação, que classifica cada dado de entrada com uma determinada classe ou categoria. Esse tipo de modelo foi o utilizado nesse projeto, pois o objetivo é justamente classificar cada um dos *pixels* da imagem de entrada como pertencente a uma das regiões da pele.

Para realizar o treinamento de um algoritmo de Aprendizado de Máquina, é importante realizar a divisão do conjunto de dados em dois subconjuntos: o de treinamento e o de teste. O primeiro é constituído da maior parte dos dados e é utilizado para treinar o algoritmo. O segundo é utilizado para que a performance do modelo seja conhecida. Se o modelo for muito robusto para a quantidade de dados disponíveis ou se a qualidade dos dados não for adequada, pode ocorrer o efeito de *overfitting*, em que o modelo aprende a classificar muito bem os dados do treinamento, mas não consegue generalizar para outros tipos de dados. O efeito contrário, chamado de *underfitting*, pode ocorrer se o modelo for muito simples comparado à complexidade dos dados de entrada. A figura 3 ilustra esses efeitos.

Figura 3 – Ilustração dos efeitos de *overfitting* e *underfitting*



Fonte: Do próprio autor

Os modelos de Aprendizado de Máquina possuem algumas variáveis que determinam como eles serão constituídos ou treinados. Essas variáveis são chamadas de hiperparâmetros e variam para cada tipo de modelo. Esses valores são escolhidos antes do treinamento

o ajuste deles é importante para otimizar o algoritmo de treinamento (GERON, 2017). Algumas medidas de *performance* são utilizadas para verificar se o modelo foi bem treinado. Para esse trabalho, a medida de performance utilizada foi a taxa de acerto, que corresponde à porcentagem de *pixels* classificados corretamente pelo modelo. A equação 2.1 mostra como essa taxa de acerto é calculada.

$$Taxa\ de\ acerto = \frac{1}{n}\sum_{i=0}^{n-1} 1(\hat{y}_i = y_i) \tag{2.1}$$

Na equação 2.1, n representa o número total de *pixels* fornecido para o modelo, y_i representa a classificação correta de um determinado *pixel* e $\hat{y}_i$ representa a classificação do *pixel* realizada pelo modelo. A função dentro do somatório é definida na expressão a seguir:

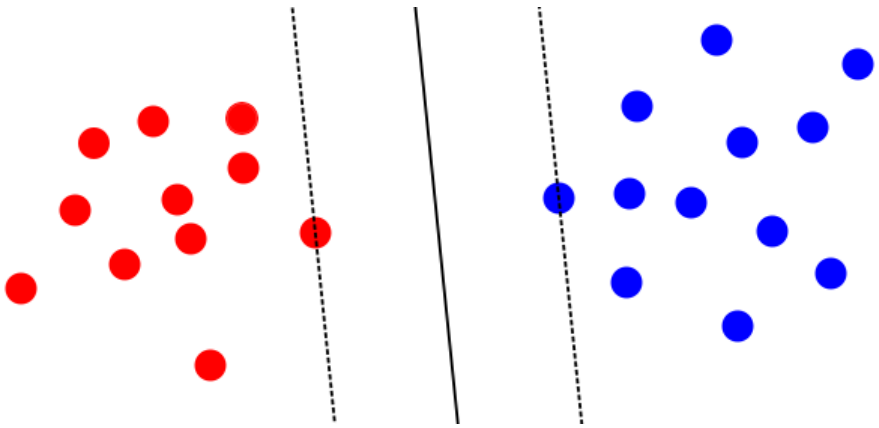
$$1(\hat{y}_i = y_i) = \begin{cases} 1, & \hat{y}_i = y_i \\ 0, & \hat{y}_i \neq y_i \end{cases}$$

Um dos modelos mais simples de classificação é o dos K vizinhos mais próximos (em inglês *K-Nearest Neighbors* ou KNN). Basicamente, o modelo classifica os dados de acordo com a proximidade entre eles. Assim, os dados desconhecidos recebem a mesma classificação do dado mais próximo conhecido pelo algoritmo. Um hiperparâmetro desse modelo é o número de vizinhos, que diz em quantos grupos os dados devem ser separados. Esse é um algoritmo simples mas que pode apresentar bons resultados (MITCHELL, 1997).

O algoritmo denominado Máquina de Vetores de Suporte (em inglês *Support Vector Machine* ou SVM) é outro exemplo de algoritmo de classificação. Esse modelo procura formar uma separação entre dados de classes diferentes com bases em amostras que estão mais próximas da fronteira, chamadas de vetores de suporte. A figura 4 mostra como esse método funciona. As amostras de duas classes distintas são representadas pelos círculos de cores vermelha e azul. As linhas tracejadas representam as divisões criadas pelas amostras da extremidade, os vetores de suporte, que formam uma margem de separação. E a linha contínua no centro representa a separação criada pelo algoritmo para classificar os dados (GERON, 2017).

Esse algoritmo é chamado de SVM linear quando utiliza funções lineares (retas ou planos) para determinar a separação entre as classes. O principal hiperparâmetro desse modelo é uma constante chamada C, que determina o quão rígido o modelo deve ser para separar os dados. Se esse hiperparâmetro tiver um valor muito alto, a margem da separação será muito pequena, o que pode diminuir o aproveitamento do modelo dados não vistos anteriormente. Se o valor de C for muito baixo, a margem fica maior e o modelo comete mais erros na classificação dos dados (GERON, 2017).

Figura 4 – Representação da separação de dados de classes distintas realizada pelo algoritmo Máquina de Vetores de Suporte



Fonte: Do próprio autor

É possível também utilizar uma função exponencial (Radial Basis Function ou RBF) para transformar os dados de entrada de forma que a distribuição das amostras seja alterada e facilite a separação entre as classes. Essa função de transformação constitui o *kernel* do algoritmo e foi utilizado nesse trabalho, referenciado por SVM RBF. Por se tratar de uma função não-linear, possui um custo computacional maior, mas geralmente apresenta resultados melhores (GERON, 2017). Além de possuir o hiperparâmetro C, esse modelo também é definido por um valor chamado *gamma*. Basicamente, esse hiperparâmetro influencia na transformação dos dados, deixando-os mais concentrados ou dispersos. Assim, escolher um *gamma* alto faz os dados se concentrarem mais e aumentar a chance de ocorrer *overfitting*.

Outro algoritmo de Aprendizado de Máquina que é utilizado para classificação é a árvore de decisão. Esse modelo é constituído basicamente de nós e ramos, que testam diversas características dos dados, separando-os conforme a profundidade da árvore aumenta. No último nível, em que os nós são chamados de folhas, os dados devem estar separados por classes. Assim, quando um novo dado passa pela árvore de decisão, as condições são testadas até que esse dado chegue em uma das folhas e tenha uma classe atribuída a ele (GERON, 2017).

Por fim, existe uma técnica de Aprendizado de Máquina chamada de *Ensemble Learning*. Nesse método, um modelo é formado a partir de outros modelos clássicos e o resultado é alcançado a partir da saída de cada um dos modelos mais simples. Um dos modelos que utiliza essa técnica é o *Random Forest* (RF), que é constituído por diversas árvores de decisões e o resultado é gerado pela moda amostral das respostas de cada árvore de decisão. Isso torna o *Random Forest* um modelo robusto e com capacidade de aprender a classificar dados complexos (GERON, 2017). O principal hiperparâmetro desse modelo é o número de árvores utilizadas na construção. Se esse número for muito alto, pode ocorrer

o *overfitting*.

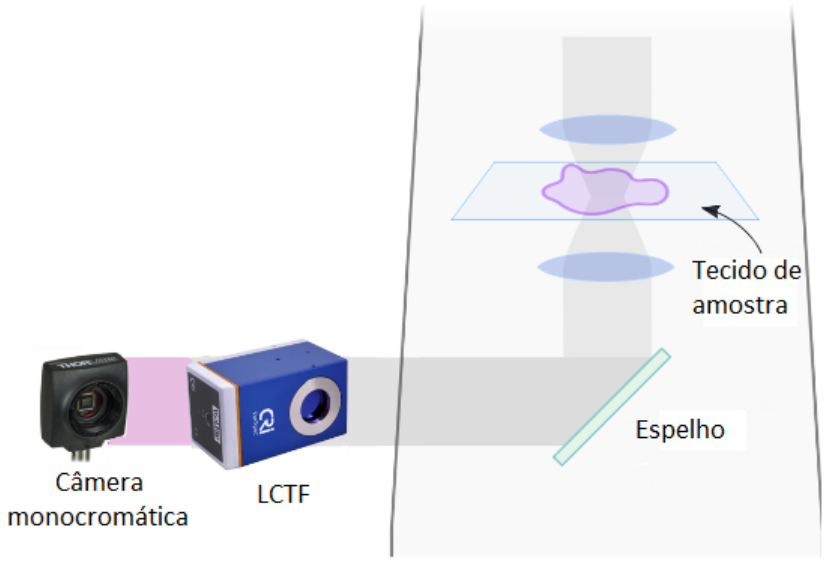
Nesse trabalho foram testados quatro modelos diferentes: o KNN, que é o mais simples, o SVM linear, o SVM com kernel RBF e o *Random Forest*, que é o mais complexo. Todos esses são modelos que já foram utilizados em trabalhos anteriores para esse tipo de aplicação (LU; FEI, 2014), (HERMES *et al.*, 2018), (ORTEGA *et al.*, 2020).

3 METODOLOGIA

3.1 Instrumentação para a aquisição das imagens

Para a aquisição das imagens, a lâmina histológica foi posicionada no microscópio óptico (Eclipse Ti-S, Nikon, Japão). O microscópio possui uma lâmpada halógena de 100 W e 12 V, que apresenta energia luminosa em todo o espectro visível (luz branca). Essa luz é focalizada na lâmina e a fração que atravessa a amostra é colimada e direcionada para uma das saídas laterais do microscópio, na qual foram acoplados um filtro ajustável eletronicamente (VIS, PerkinElmer, EUA) e uma câmera monocromática (DCC-1545M, Thorlabs, EUA). A câmera e a LCTF foram conectadas ao computador via cabo USB para controle via LabVIEW. A figura 5 mostra um diagrama que ilustra como a instrumentação do sistema foi disposta e a figura 6 mostra uma foto do microscópio utilizado, juntamente à câmera e à LCTF acopladas.

Figura 5 – Diagrama da instrumentação do projeto com a ilustração do caminho percorrido pela luz, que passa pela amostra, pelo espelho, pela LCTF e chega até a câmera monocromática



Fonte: Do próprio autor

3.2 Algoritmo para a aquisição das imagens multiespectrais

Um dos alunos que participa do mesmo grupo de pesquisa desenvolveu um algoritmo em LabVIEW para o controle da câmera e da LCTF. De forma resumida, o algoritmo inicializa a câmera e a LCTF usando arquivos de configuração fornecidos pelos respectivos fabricantes. Em seguida, uma rotina de aquisição é definida. Essa rotina é caracterizada

Figura 6 – Foto do microscópio e do acoplamento da câmera monocromática e da LCTF

Fonte: Do próprio autor

pelos valores inicial e final do comprimento de onda e pelo incremento. Quando a rotina é inicializada, o computador envia o comprimento de onda que será capturado para a LCTF e configura a câmera ajustando o tempo de exposição, a taxa de quadros e o *pixel clock*. Após isso, a imagem é adquirida e salva em uma determinada pasta do computador. O comprimento de onda é incrementado e todos esses procedimentos são repetidos. O processo ocorre em *loop* até que todos os comprimentos de onda sejam adquiridos.

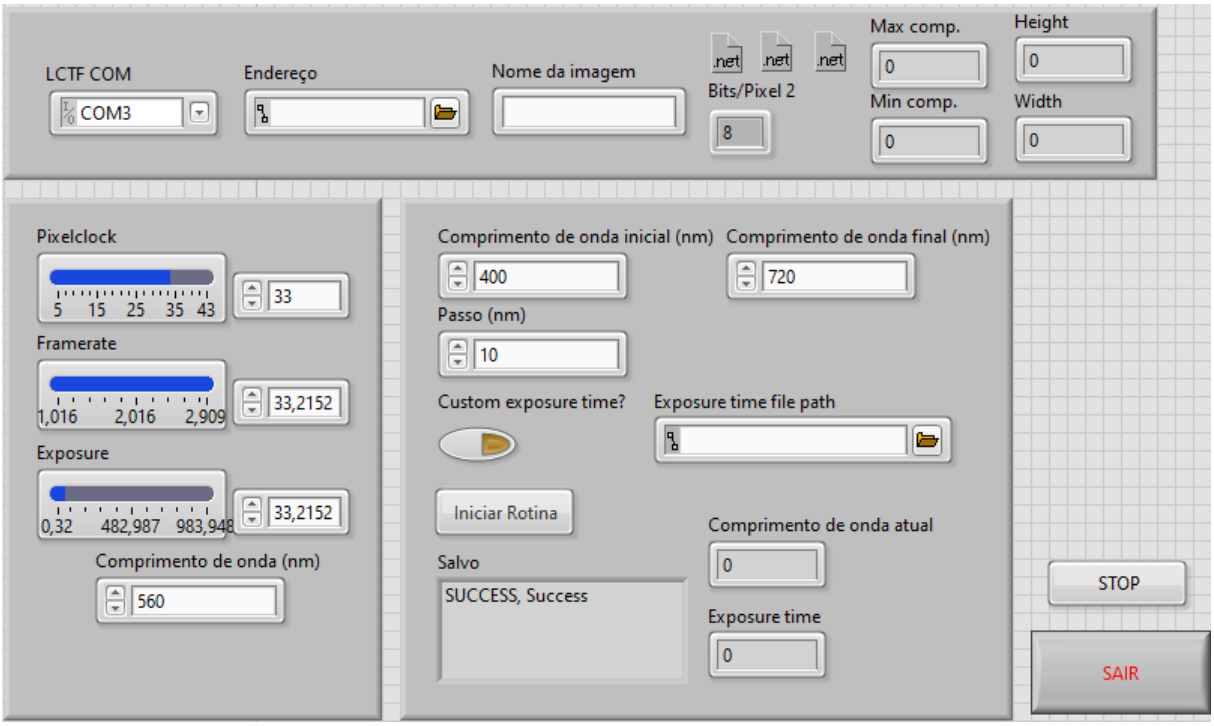
O resultado final é uma pasta com n imagens monocromáticas, em que n é o número de comprimentos de onda adquiridos. A figura 7 mostra a interface do usuário com o algoritmo desenvolvido em LabVIEW.

As imagens geradas possuem 1280 x 1024 *pixels* e os comprimentos de onda adquiridos foram de 400 a 720 nm, com incremento de 10 nm, o que resultou em 33 imagens monocromáticas. Essas imagens foram agrupadas usando um algoritmo em Python, que gerou a imagem multiespectral. A imagem RGB correpondente foi criada a partir das imagens que representam os comprimentos de onda de 470, 570 e 630 nm.

3.3 Pré-processamento e preparação dos dados

Como pré-processamento das imagens, foi realizada uma normalização pela região do vidro da lâmina como forma de obter a transmitância da imagem. Para realizar esse

Figura 7 – Interface do algoritmo implementado em LabVIEW, com diversos parâmetros de controle e com a imagem adquirida sendo apresentada na tela



Fonte: Do próprio autor

processo, foi selecionada uma região retangular da imagem que corresponde à região do vidro e na qual não há nenhum tecido ou célula presente. Em seguida, foi calculada a média da intensidade dos *pixels* presentes nessa região para cada comprimento de onda. Por fim, todos os *pixels* da imagem foram divididos por essa média, para cada comprimento de onda. Essa normalização reduz a diferença entre lâminas diferentes e que foram imageadas em condições distintas, como por exemplo a luminosidade. Além disso, esse processo de normalização fornece o índice de transmitância para cada pixel da imagem.

Além da normalização pela região do vidro, houve uma normalização completa da imagem, de forma que a intensidade de todos os *pixels* estivesse entre 0 e 1. Esse procedimento é necessário pois alguns modelos de Aprendizado de Máquina são prejudicados se os dados não estiverem normalizados (GERON, 2017). Pode-se citar os modelos KNN e SVM, que são influenciados pela distribuição de valores dos dados de entrada.

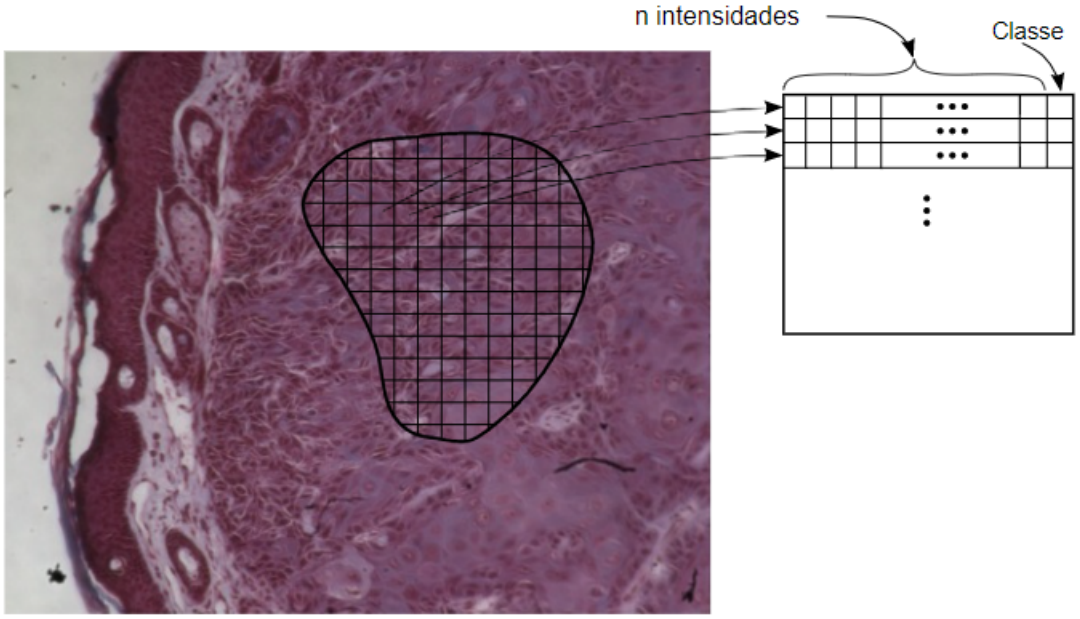
Após o processo de normalização, o filtro passa-baixa foi aplicado em todos os canais da imagem. A máscara utilizada teve dimensão de 20 x 20 *pixels*. As imagens e os dados foram manipulados e processados através de funções das seguintes bibliotecas de código aberto em *Python*: *numpy* (HARRIS *et al.*, 2020), *opencv* (BRADSKI, 2000) e *matplotlib* (HUNTER, 2007).

3.4 Treinamento, teste e armazenamento dos modelos

Os algoritmos de classificação utilizados nesse trabalho são supervisionados. Portanto, é necessário fornecer um conjunto de dados junto com as classes de saída. Para isso, foi criada uma rotina que permite selecionar as regiões da imagem correspondente a cada classe, a qual é representada por um número inteiro.

Essa rotina funciona da seguinte forma: inicialmente, uma região poligonal é selecionada na imagem e os índices dos *pixels* que estão dentro dessa região são armazenados em uma variável. Em seguida, os índices são acessados de um em um para obter o valor de intensidade de cada comprimento de onda do *pixel* correspondente, o que gera um vetor de intensidades. Ao final desse vetor é acrescentado o número da classe (região) à qual o *pixel* pertence. Os vetores correspondentes à cada *pixel* são anexados um após o outro, formando um matriz em que as *n-1* primeiras colunas são os valores dos *pixels*, a última coluna possui informação da classe e as linhas são os próprios *pixels*. A figura 8 ilustra de forma simplificada como essa matriz é formada.

Figura 8 – Processo de escolha das regiões do tecido e separação dos pixels



Fonte: Do próprio autor

Essa rotina de seleção das regiões ocorre com quantas imagens estiverem disponíveis para o treinamento e os *pixels* classificados pelo usuário são todos anexados na mesma variável. A quantidade de dados de entrada é definida pela quantidade total de pixels selecionados para o treinamento. Dessa forma, poucas imagens são suficientes para treinar o modelo, pois uma imagem pode gerar uma matriz com mais de um milhão de linhas. Cada imagem é composta de 1280 x 1024 *pixels*, o que corresponde a 1.310.720 elementos. Para um modelo de Aprendizado de Máquina, quanto mais dados, melhor, pois assim o

algoritmo adquire maior capacidade de generalização e consegue classificar dados diferentes corretamente.

Uma patologista auxiliou na identificação das regiões da lâmina para o treinamento do modelo. Com esse procedimento realizado, os dados foram divididos aleatoriamente entre dois conjuntos de dados: o de treinamento, composto por 80% dos dados, e o de teste, formado pelos 20% restantes.

O conjunto de treinamento foi, então, aplicado na entrada de quatro algoritmos de Aprendizado de Máquina. O primeiro foi o KNN, com o número de vizinhos igual ao número de classes escolhidas (5 classes). O segundo foi o SVM linear, com o parâmetro C igual a 1. O terceiro foi o SVM com *kernel* RBF, com C igual a 1 e *gamma* igual a 2. Por fim, o último modelo foi o *Random Forest*, com 100 árvores. A biblioteca de código aberta em *Python* utilizada para gerar e treinar os modelos foi a *scikit-learn* (BUITINCK *et al.*, 2013). O tempo do treinamento de cada modelo foi registrado para análise posterior.

Após o modelo ser treinado, o conjunto de teste foi utilizado para que a taxa de acerto do modelo fosse calculada. Essa função já está implementada internamente nos modelos da biblioteca *scikit-learn*.

Após os modelos estarem treinados, eles foram armazenados em arquivos externos que foram salvos em uma determinada pasta no computador. Dessa forma, os modelos podem ser acessados a qualquer momento sem precisarem passar pela etapa do treinamento novamente. A biblioteca em *Python* utilizada para salvar e carregar os modelos treinados foi a *Pickle* (ROSSUM, 2020).

O treinamento e a classificação do algoritmo foram realizados duas vezes. Na primeira vez as imagens foram utilizadas sem nenhum filtro e, na segunda vez, as imagens de entrada passaram por um filtro passa-baixa.

3.5 Classificação e visualização do resultado

Algumas imagens foram separadas para serem classificas pelos modelos treinados. Para isso, elas passaram pelo processo de normalização pela região do vidro e para que os valores de intensidade dos *pixels* ficasse entre 0 e 1. Em seguida, elas foram transformadas de matriz para vetor, de forma que os 33 canais correspondentes às intensidades dos diversos comprimentos de onda se tornassem colunas e todos os *pixels* se tornassem linhas. Esses processos foram realizados da mesma forma que ocorreram para as imagens de treinamento, de forma que as condições entre as imagens fossem as mesmas e os modelos pudessem classificar com eficiência.

A matriz resultante foi aplicada ao modelo, que retornou um vetor com o mesmo número de linhas da matriz de entrada e com apenas uma coluna, que contém um número inteiro representante da classe atribuída a cada *pixel*. Portanto, o algoritmo desenvolvido

classifica a imagem *pixel* a *pixel*, atribuindo a cada um deles uma região específica. Para recuperar a imagem, o vetor-coluna resultante foi reformatado para o formato da imagem original e um algoritmo de coloração atribuiu uma cor diferente para cada *pixel*, de acordo com a sua classe.

Dessa forma, foi possível obter as imagens classificadas e com cada região de uma cor diferente, para visualização. Após isso, foi criado um algoritmo, semelhante ao do treinamento, que permitiu a classificação manual das regiões das imagens de classificação, a fim de comparar com os resultados gerados pelos modelos e calcular a taxa de acerto. Esse procedimento foi importante para fornecer um valor de taxa de acerto mais fidedigno em relação à capacidade do algoritmo de generalizar o aprendizado para dados diferentes dos que foram utilizados no treinamento.

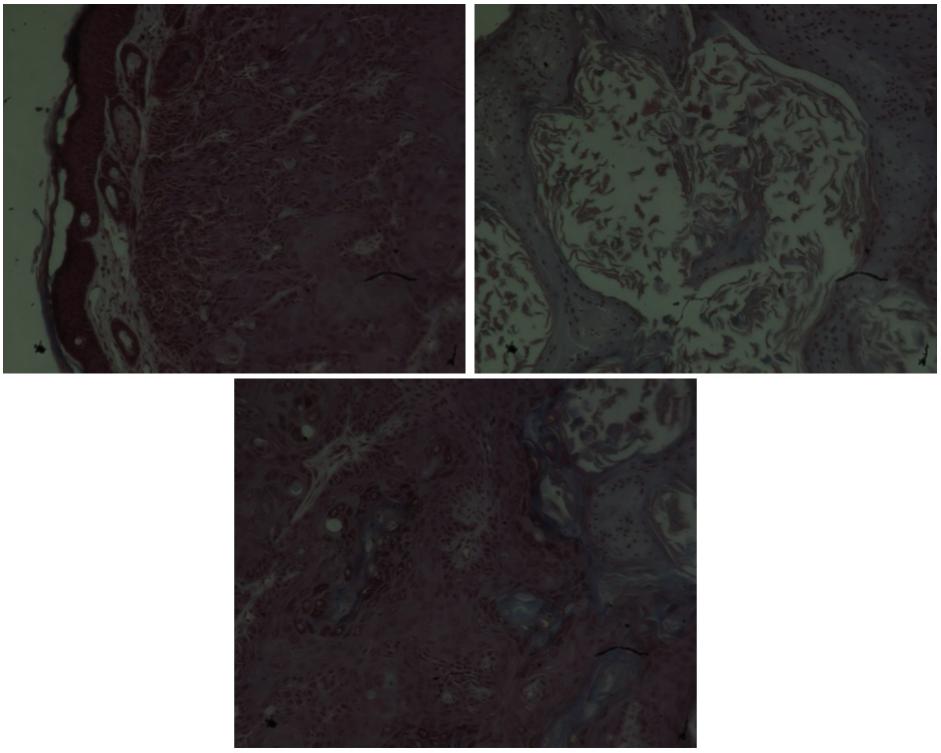
Como pós-processamento, foi aplicado o filtro da mediana com uma máscara de dimensão 15 x 15 *pixels* para suavizar a imagem e melhorar a taxa de acerto. O código-fonte do algoritmo é apresentado no apêndice A.

4 RESULTADOS E DISCUSSÕES

4.1 Aquisição das imagens

Utilizando o algoritmo desenvolvido em LabVIEW, foram obtidas 5 imagens multiespectrais, cujas imagens RGB correspondentes são apresentadas nas figuras 9 e 10.

Figura 9 – Apresentação das três imagens RGB extraídas das imagens multiespectrais adquiridas para o treinamento



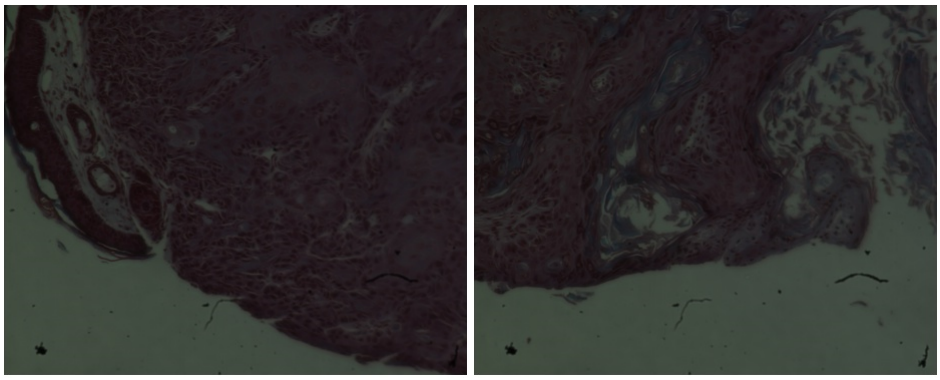
Fonte: Do próprio autor

As três primeiras imagens foram utilizadas no treinamento, enquanto que as duas últimas foram utilizadas na classificação.

4.2 Normalização pela região do vidro

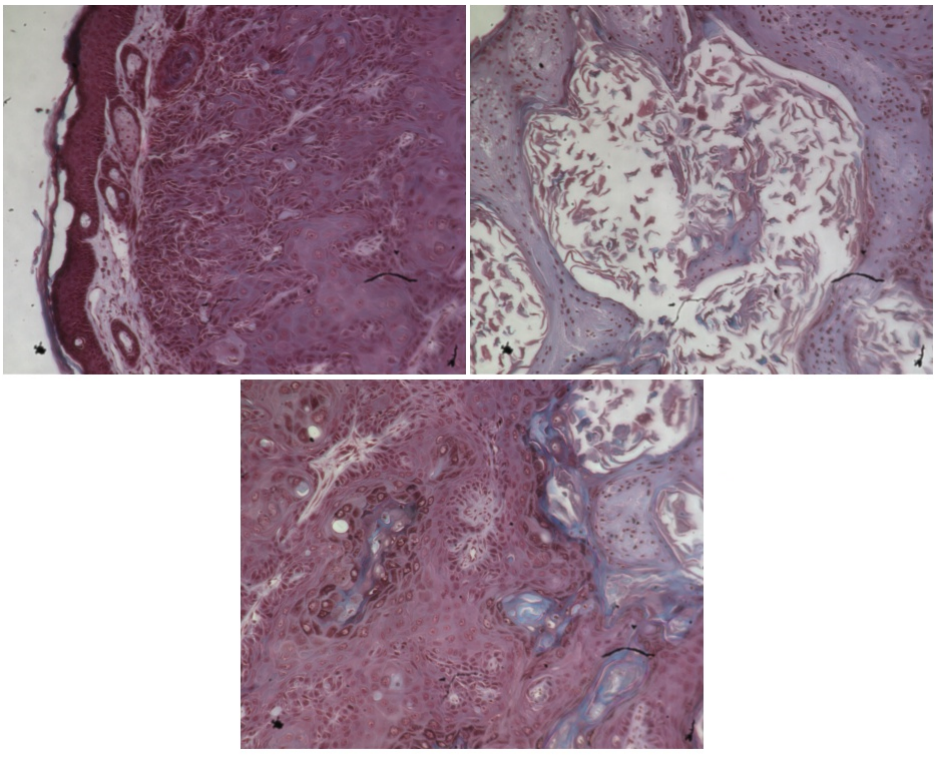
A normalização pela região do vidro foi realizada para se obter o índice da transmitância em cada pixel e diminuir fatores de variabilidade, como luminosidade e tempo de exposição da câmera. Após esse processo de normalização, as imagens apresentaram maior contraste e coloração, conforme apresentado nas figuras 11 e 12.

Figura 10 – Apresentação das três imagens RGB extraídas das imagens multiespectrais adquiridas para a classificação



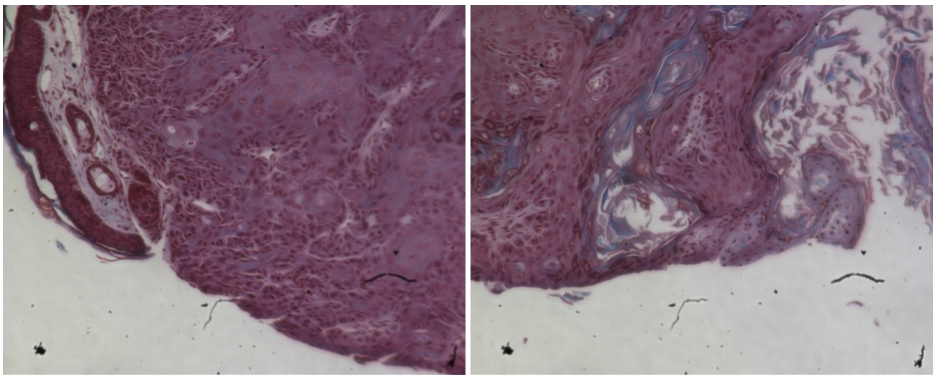
Fonte: Do próprio autor

Figura 11 – Imagens RGB resultantes após o processo de normalização pela região do vidro das imagens utilizadas no treinamento dos modelos



Fonte: Do próprio autor

Figura 12 – Imagens RGB resultantes após o processo de normalização pela região do vidro das imagens separadas para a classificação pelos modelos



Fonte: Do próprio autor

4.3 Sem aplicação do filtro passa-baixa

Os resultados apresentados nessa seção mostram o treinamento, o teste e a classificação das imagens sem a aplicação do filtro passa-baixa.

4.3.1 Treinamento dos modelos

Após o treinamento dos quatro modelos de Aprendizado de Máquina, os tempos de treinamento foram registrados e são apresentados na tabela 1.

Tabela 1 – Tempo necessário para o processo de treinamento dos modelos

Modelo	Tempo de treinamento (s)
KNN	1542
SVM linear	142
SVM RBF	3715
Random Forest	544

O tempo de treinamento pode ser utilizado como um critério secundário para a escolha do algoritmo a ser utilizado. O parâmetro mais importante é a taxa de acerto, porém deve-se avaliar o custo computacional caso os modelos apresentem valores de taxa de acerto semelhantes.

4.3.2 Teste dos modelos

O teste realizado permitiu calcular a taxa de acerto de cada modelo utilizando o conjunto de teste previamente separado. A tabela 2 mostra o resultado para esse teste.

Todos os modelos apresentaram valores bem altos de taxa de acerto, porém os dados de teste utilizados foram extraídos das mesmas imagens utilizadas para extrair os

Tabela 2 – Taxa de acerto para os quatro modelos treinados utilizando o conjunto de testes como base para o cálculo

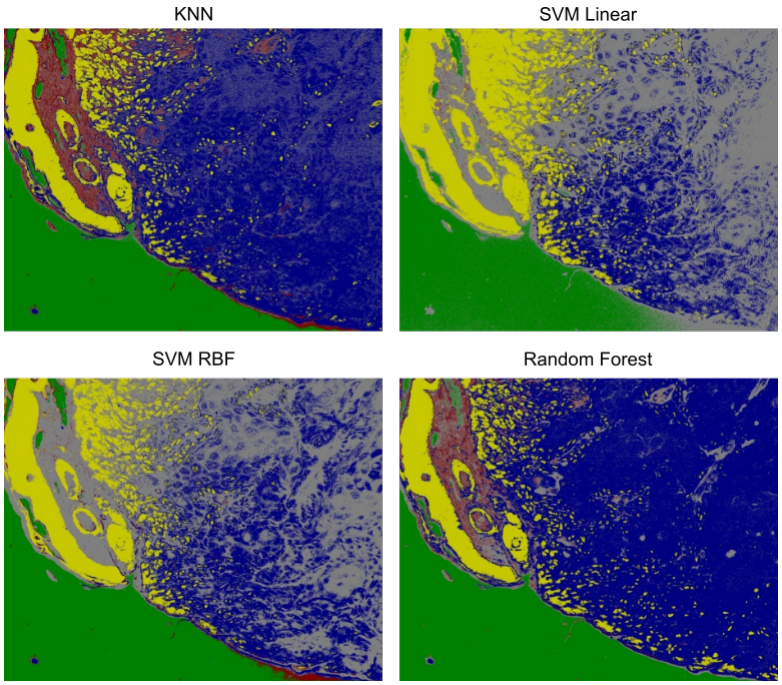
Modelo	Taxa de acerto
KNN	0,991
SVM linear	0,978
SVM RBF	0,992
Random Forest	0,991

dados do treinamento. Assim, não é possível concluir que o modelo está generalizando bem para outras imagens. Para a classificação, foram usadas imagens não vistas pelo algoritmo no treinamento.

4.3.3 Imagens classificadas

As duas imagens apresentadas na figura 10 foram classificadas pelos quatro modelos e coloridas artificialmente para facilitar a visualização das classes. As figuras 13 e 14 mostram os resultados.

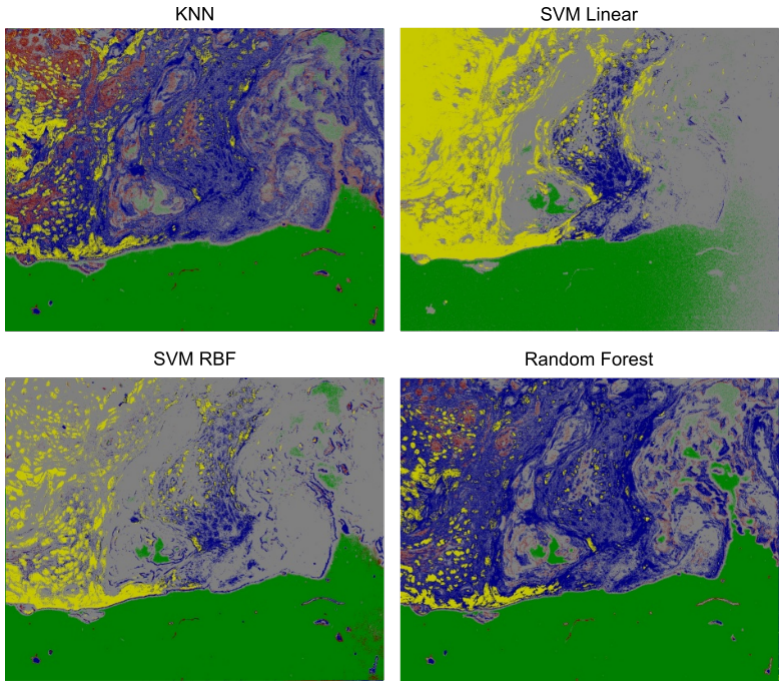
Figura 13 – Resultado da classificação da primeira imagem pelos quatro algoritmos e com coloração artificial para representação das regiões



Fonte: Do próprio autor

A cor vermelha representa a derme, a amarela corresponde à epiderme, a verde mostra a região do vidro e as cores azul e cinza indicam a região do tumor. Duas classes foram utilizadas para essa última região porque os tecidos eram diferentes.

Figura 14 – Resultado da classificação da segunda imagem pelos quatro algoritmos e com coloração artificial para representação das regiões



Fonte: Do próprio autor

Percebe-se que a classificação da segunda imagem não apresentou regiões muito uniformes. Essa imagem mostra um pedaço da lâmina em que aparecem diversas estruturas que não foram treinadas para serem reconhecidas pelo algoritmo. Por isso ocorre a ausência de regiões uniformes.

A taxa de acerto dos modelos também foi calculada para as duas imagens classificadas. Os resultados são apresentados na tabela 3.

Tabela 3 – Taxa de acerto dos modelos treinados para as imagens classificadas

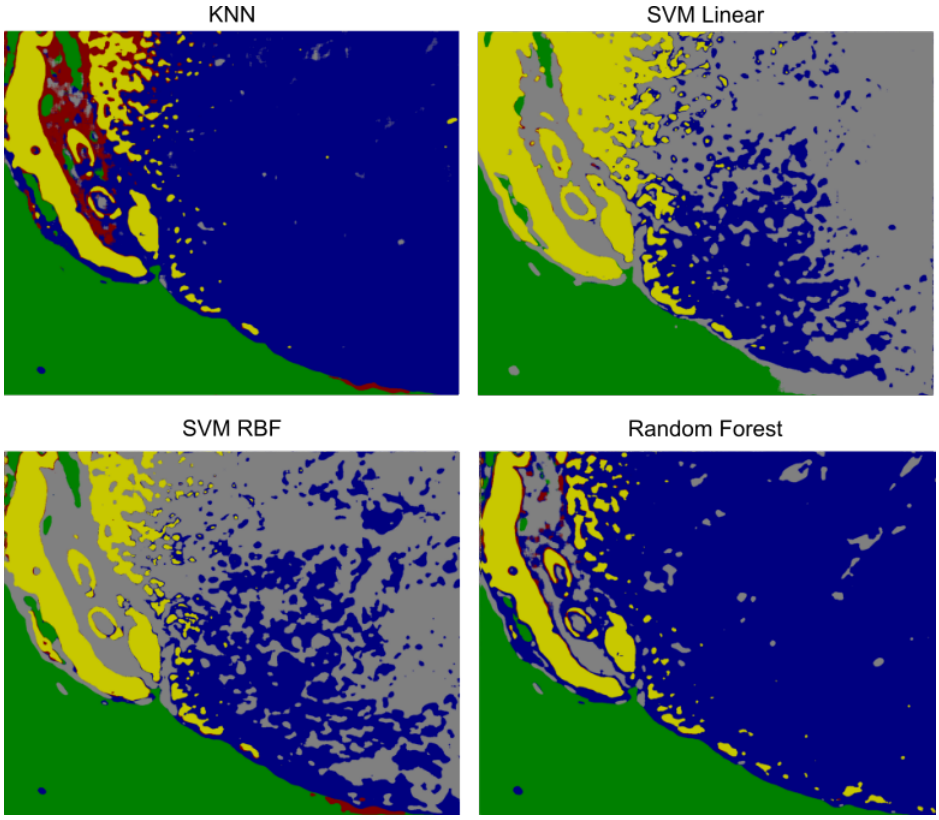
Modelo	Taxa de acerto da imagem 1	Taxa de acerto da imagem 2
KNN	0,88	0,84
SVM linear	0,48	0,62
SVM RBF	0,69	0,65
Random Forest	0,85	0,84

Os dados de taxa de acerto confirmam que a segunda imagem teve uma classificação um pouco pior para a maioria dos modelos. No entanto, a diferença foi menor que 0,04 nos casos em que houve piora. Portanto, os algoritmos foram consistentes nas duas imagens. Os melhores modelos foram o KNN e o *Random Forest*, com uma taxa de acerto relativamente boa, considerando que nenhum pré-processamento foi aplicado (MAADEED *et al.*, 2017).

4.3.4 Aplicação do filtro da mediana

O filtro da mediana foi aplicado para as duas imagens classificadas. Os resultados são apresentados nas figuras 15 e 16. Além disso, a taxa de acerto dos modelos também foi calculada após a aplicação do filtro da mediana (tabela 4).

Figura 15 – Resultado da classificação da primeira imagem após aplicação do filtro da mediana



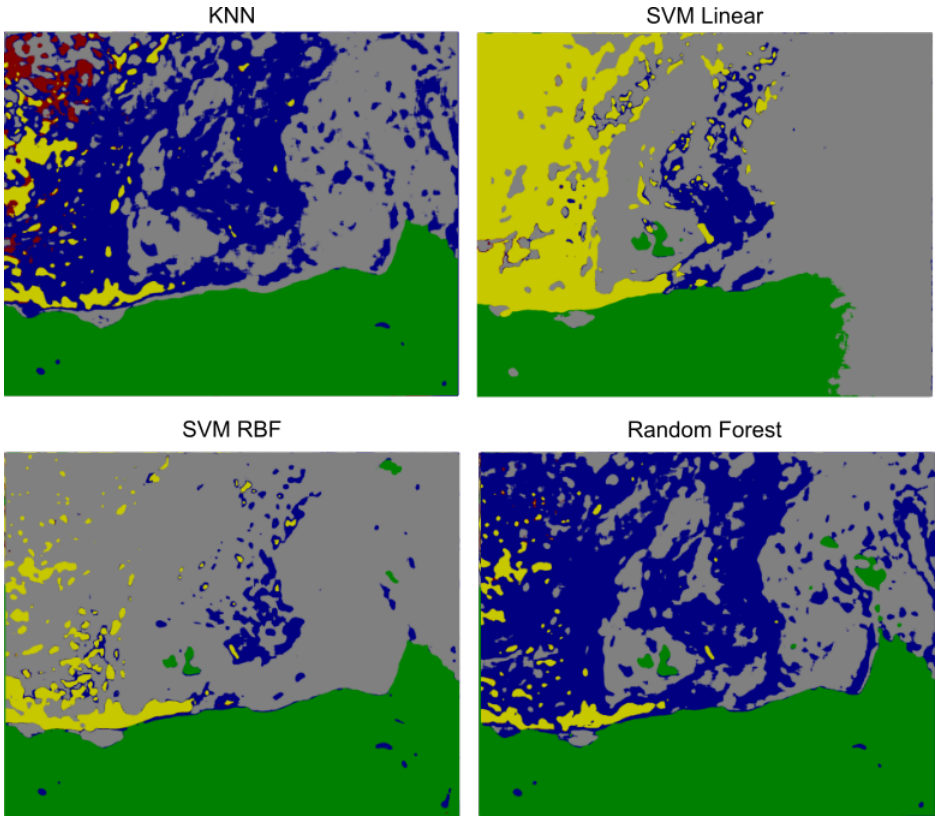
Fonte: Do próprio autor

Tabela 4 – Taxa de acerto dos modelos para as duas imagens após a aplicação do filtro da mediana como pós-processamento

Modelo	Taxa de acerto da imagem 1	Taxa de acerto da imagem 2
KNN	0,94	0,90
SVM linear	0,55	0,68
SVM RBF	0,79	0,65
Random Forest	0,93	0,89

Comparando as tabelas 3 e 4, nota-se que o filtro da mediana aumentou a taxa de acerto de todos os modelos. Novamente, os modelos KNN e *Random Forest* apresentaram os melhores resultados. Os modelos apresentaram maior discrepância na taxa de acerto entre a primeira e a segunda imagem após a aplicação da mediana.

Figura 16 – Resultado da classificação da segunda imagem após aplicação do filtro da mediana



Fonte: Do próprio autor

4.4 Com aplicação do filtro passa-baixa

A aplicação do filtro passa-baixa gerou as imagens apresentadas nas figuras 17 e 18. Conforme é o objetivo do filtro, as imagens foram suavizadas e apresentam aspecto embaçado. Dessa forma, as estruturas menores têm menor influência nos modelos.

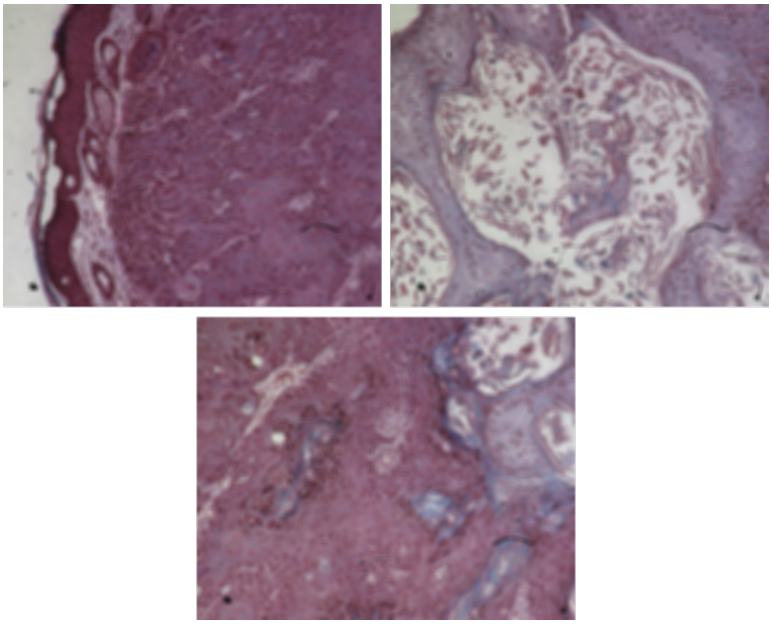
4.4.1 Treinamento dos modelos

A tabela 5 apresenta os tempos de execução do treinamento dos modelos para as imagens filtradas.

Tabela 5 – Tempo necessário para o treinamento dos modelos

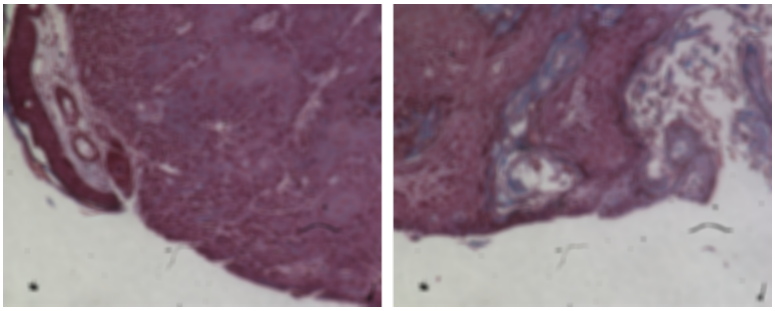
Modelo	Tempo de treinamento (s)
KNN	28
SVM linear	79
SVM RBF	2501
Random Forest	632

Figura 17 – Imagens separadas para o treinamento dos modelos filtradas com o filtro passa-baixa



Fonte: Do próprio autor

Figura 18 – Imagens separadas para a classificação pelos modelos filtradas com o filtro passa-baixa



Fonte: Do próprio autor

Comparando com os tempos de treinamento para as imagens não filtradas, houve redução para todos os modelos, exceto para o *Random Forest*, que passou de 544 para 632 s.

4.4.2 Teste dos modelos

A tabela 6 apresenta os valores de taxa de acerto calculados para o conjunto de teste para as imagens filtradas.

A taxa de acerto dos modelos apresentou aumento quando se utilizou as imagens filtradas. Alguns modelos alcançaram 100% de taxa de acerto, porém isso mostra apenas que os modelos classificam bem os dados parecidos com os quais eles foram treinados. É

Tabela 6 – Taxa de acerto para os modelos treinados com imagens filtradas

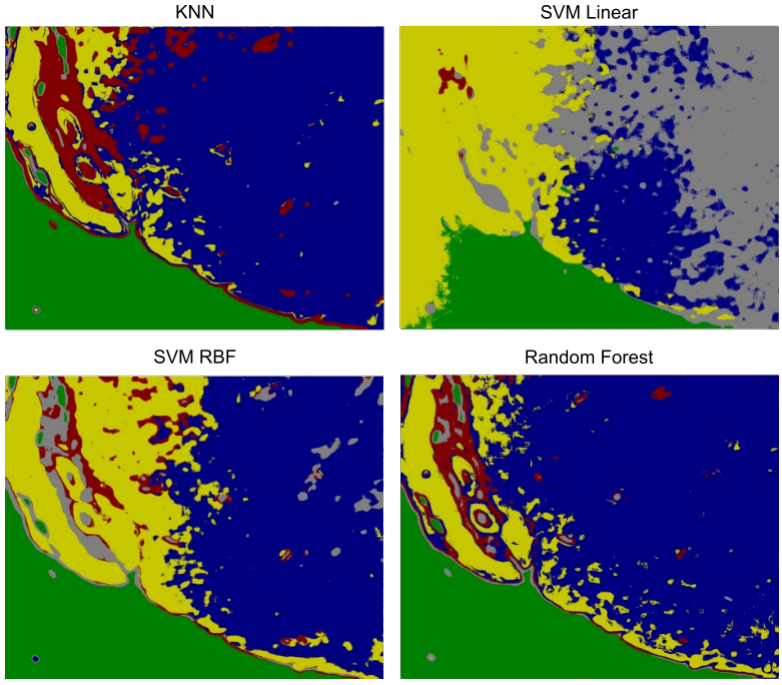
Modelo	Taxa de acerto
KNN	1
SVM linear	0,997
SVM RBF	1
Random Forest	1

importante medir a taxa de acerto para dados diferentes, o que foi realizado posteriormente.

4.4.3 Imagens classificadas

As figuras 19 e 20 mostram os resultados da classificação das imagens com filtro passa-baixa.

Figura 19 – Resultado da classificação da primeira imagem com filtro passa-baixa

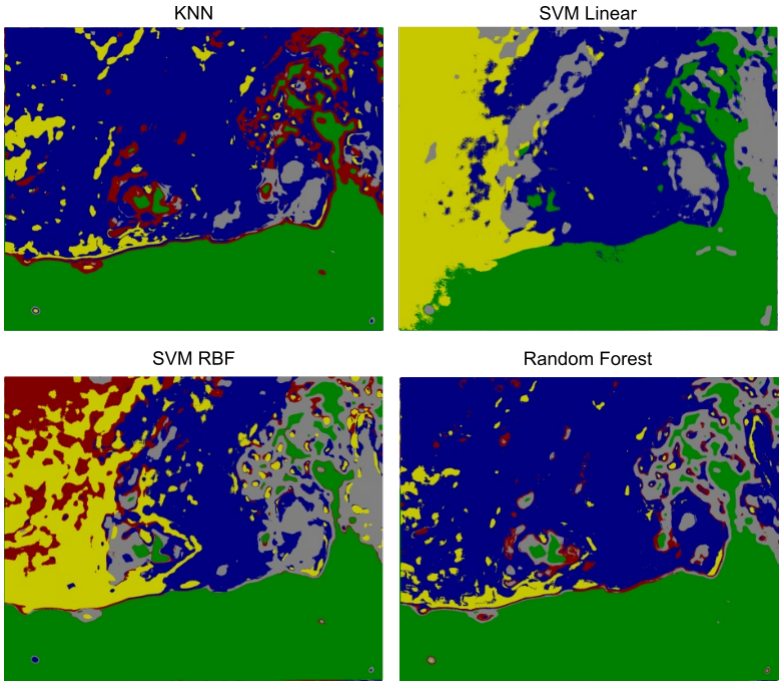


Fonte: Do próprio autor

O modelo SVM Linear não teve um aproveitamento bom para essas imagens filtradas, porém os outros modelos apresentaram uniformidade maior nas regiões, resultado do filtro passa-baixa. A tabela 7 mostra a taxa de acerto dos modelos para essas imagens.

A aplicação do filtro passa-baixa foi relevante para aumentar a taxa de acerto da classificação. Mesmo sem o uso do filtro da mediana após a classificação, houve um aumento considerável em relação ao uso das imagens sem pré-processamento.

Figura 20 – Resultado da classificação da segunda imagem com filtro passa-baixa



Fonte: Do próprio autor

Tabela 7 – Taxa de acerto dos modelos treinados para as imagens filtradas

Modelo	Taxa de acerto da imagem 1	Taxa de acerto da imagem 2
KNN	0,92	0,93
SVM linear	0,45	0,67
SVM RBF	0,77	0,68
Random Forest	0,93	0,92

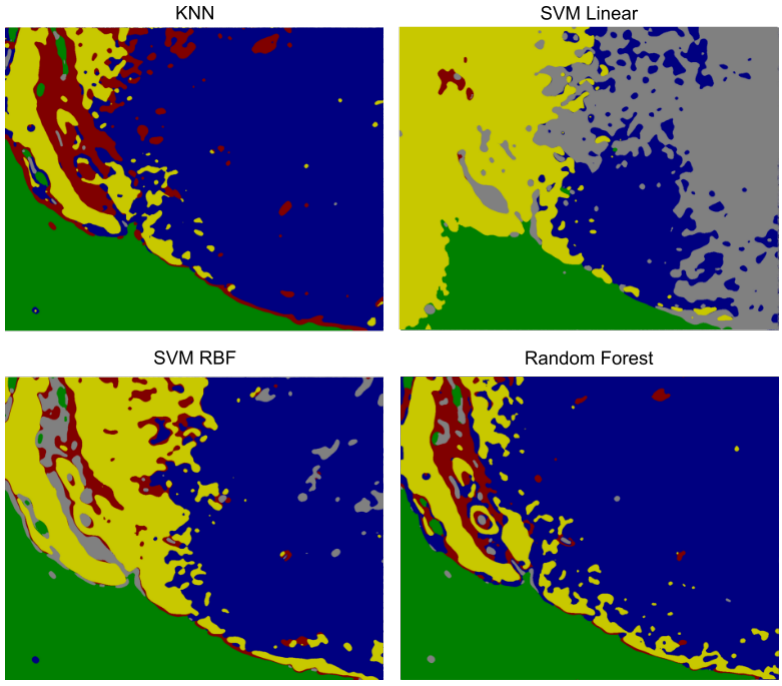
4.4.4 Aplicação do filtro da mediana

As figuras 21 e 22 mostram o resultado da aplicação do filtro da mediana nas imagens classificadas.

Nota-se nas imagens resultantes que o filtro da mediana apenas suavizou mais as fronteiras das regiões das imagens. Mas houve menos alteração porque o filtro passa-baixa já tinha causado um efeito semelhante. A tabela 8 apresenta os valores de taxa de acerto dos modelos após a aplicação do filtro da mediana.

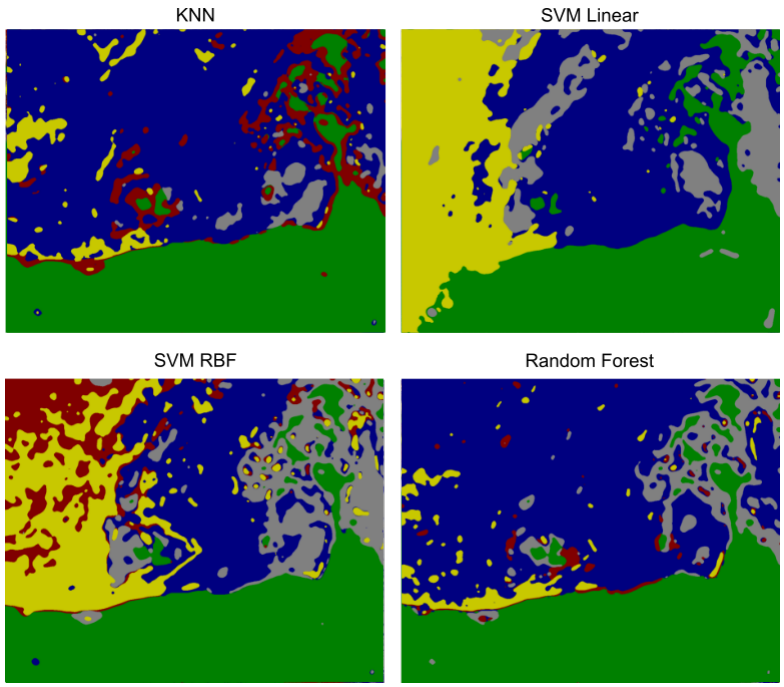
O filtro da mediana novamente teve importância no aumento da taxa de acerto dos modelos para as duas imagens classificadas. Os valores de taxa de acerto dos modelos KNN e *Random Forest* são bons se comparados a outros modelos descritos na literatura (MAADEED *et al.*, 2017).

Figura 21 – Resultado da classificação da primeira imagem após aplicação do filtro da mediana



Fonte: Do próprio autor

Figura 22 – Resultado da classificação da segunda imagem após aplicação do filtro da mediana



Fonte: Do próprio autor

Tabela 8 – Taxa de acerto dos modelos treinados para as imagens filtradas com os filtros passa-baixa e mediana

Modelo	Taxa de acerto da imagem 1	Taxa de acerto da imagem 2
KNN	0,92	0,94
SVM linear	0,45	0,69
SVM RBF	0,77	0,68
Random Forest	0,94	0,92

5 CONCLUSÃO

Foi desenvolvido um algoritmo que mostrou a taxa de acerto de quatro modelos ao classificar imagens multiespectrais de lâminas histológicas. Com os resultados, foi possível concluir que os melhores modelos para essa finalidade são o KNN e o *Random Forest*, que apresentaram valor de 0,93 de taxa de acerto na média para as duas imagens filtradas e com aplicação do filtro da mediana.

Mostrou-se a importância de utilizar filtro passa-baixa como pré-processamento, que elevou a taxa de acerto dos modelos e diminuiu o tempo de treinamento. Outro processo importante foi o filtro da mediana como pós-processamento, que uniformizou as regiões e elevou a taxa de acerto final de todos os modelos.

Enfim, com esse trabalho mostrou-se que existem modelos de Aprendizado de Máquina capazes de classificar as imagens multiespectrais e auxiliar os patologistas no reconhecimento dos diversos tecidos, principalmente com a habilidade de identificar regiões de tumor.

Para futuros trabalhos, é importante adquirir mais imagens e com mais diversidade de estruturas, de forma a tornar os modelos mais completos. Além disso, pela maior quantidade e qualidade de dados o modelo poderá apresentar maior taxa de acerto na classificação de lâminas não vistas anteriormente. O modelo também pode ser treinado para reconhecer diversas outras estruturas presentes na pele de camundongo.

Por fim, um algoritmo mais elaborado pode ser desenvolvido para que informações quantitativas possam ser extraídas após a classificação da imagem, como a área da região de tumor, por exemplo. O algoritmo também pode ser utilizado para monitorar a evolução de uma certa patologia. Enfim, são diversos os caminhos que estão disponíveis após esse trabalho inicial.

REFERÊNCIAS

BRADSKI, G. The OpenCV Library. **Dr. Dobb's Journal of Software Tools**, 2000.

BUITINCK, L. *et al.* API design for machine learning software: experiences from the scikit-learn project. *In*: **ECML PKDD Workshop: Languages for Data Mining and Machine Learning**. [*S.l.: s.n.*], 2013. p. 108–122.

FARKAS, D.; BECKER, D. Applications of spectral imaging: Detection and analysis of human melanoma and its precursors. **Pigment cell research / sponsored by the European Society for Pigment Cell Research and the International Pigment Cell Society**, v. 14, p. 2–8, 03 2001.

GERON, A. **Hands-On Machine Learning with Scikit-Learn and TensorFlow**. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Inc., 2017. ISBN 978-1-491-96229-9.

GONZALEZ, R. C.; WOODS, R. E.; EDDINS, S. L. **Digital Image Processing Using MATLAB: AND "Mathworks, MATLAB Sim SV 07 "**. USA: Prentice Hall Press, 2007. ISBN 1405893281.

HARRIS, C. R. *et al.* Array programming with NumPy. **Nature**, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: https://doi.org/10.1038/s41586-020-2649-2.

HERMES, M. *et al.* Mid-ir hyperspectral imaging for label-free histopathology and cytology. **Journal of Optics**, IOP Publishing, v. 20, n. 2, p. 023002, 2018.

HUNTER, J. D. Matplotlib: A 2d graphics environment. **Computing in Science & Engineering**, IEEE COMPUTER SOC, v. 9, n. 3, p. 90–95, 2007.

LU, G.; FEI, B. Medical hyperspectral imaging: a review. **Journal of biomedical optics**, International Society for Optics and Photonics, v. 19, n. 1, p. 010901, 2014.

MAADEED, S. A. *et al.* Multispectral imaging and machine learning for automated cancer diagnosis. *In*: **2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)**. [*S.l.: s.n.*], 2017. p. 1740–1744.

MANSFIELD, J. R. Multispectral imaging: A review of its technical aspects and applications in anatomic pathology. **Veterinary Pathology**, v. 51, n. 1, p. 185–210, 2014. PMID: 24129898. Disponível em: https://doi.org/10.1177/0300985813506918.

MITCHELL, T. **Machine learning**. 1st. ed. McGraw Hill Higher Education, 1997. (Mcgraw-Hill International Edit). ISBN 9780071154673,0071154671. Disponível em: http://gen.lib.rus.ec/book/index.php?md5=b79ff895e0f7668ed1c931631a4acc9b.

ORTEGA, S. *et al.* Hyperspectral and multispectral imaging in digital and computational pathology: a systematic review. **Biomedical Optics Express**, Optical Society of America, v. 11, n. 6, p. 3195–3233, 2020.

ROSSUM, G. V. **The Python Library Reference, release 3.8.2**. [*S.l.: s.n.*]: Python Software Foundation, 2020.

APÊNDICES

APÊNDICE A – CÓDIGO-FONTE

A seguir será apresentado o código criado em *Python* para o projeto.

A.1 Bibliotecas importadas

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
import time
import os

from cv2 import medianBlur
from cv2 import blur

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import LinearSVC
from sklearn.svm import SVC

from pickle import dump
from pickle import load

import imfun
```

A biblioteca imfun foi desenvolvida para conter algumas funções utilizadas no algoritmo.

A.2 Carregamento e normalização da imagem

```
folder = '...'

train, test = imfun.open_images(folder)
w, h, d = train[0].shape

wavelength = np.linspace(400, 720, 33)
```

```
bgr = []
rgb = []

for i in range(len(train)):
    rgb_temp, bgr_temp, _ = imfun.create_rgb(train[i], wavelength)
    train[i], rgb_temp, bgr_temp = imfun.normalize_glass(train[i],
    bgr_temp, wavelength)

for i in range(len(test)):
    rgb_temp, bgr_temp, _ = imfun.create_rgb(test[i], wavelength)
    test[i], _, _ = imfun.normalize_glass(test[i], bgr_temp, wavelength)
```

A.3 Filtro passa-baixa

```
# O primeiro loop é para cobrir todas as imagens de treinamento e o
# segundo loop é para varrer todos os comprimentos de onda

for i in range(len(train)):
    for j in range(d):
        train[i][:,:,j] = blur(train[i][:,:,j], (20,20))

for i in range(len(test)):
    for j in range(d):
        test[i][:,:,j] = blur(test[i][:,:,j], (20,20))
```

A.4 Seleção das regiões manualmente para gerar os dados de treinamento e teste

```
label = []
n_labels = 0
train_data = []
keep = 'y'

while keep == 'y':
    label.append(input('Name of the region:'))
    for i in range(len(train)):
        data_temp = imfun.define_labels(train[i], bgr[i], n_labels)

        # Condição para verificar se uma região foi mesmo selecionada
```

```
        if type(data_temp) != int:
            train_data.append(data_temp)

    keep = input('Do you wish to add another region? [y/n]: ')
    n_labels += 1

data = train_data[0].copy()

for i in range(1,len(train_data)):
    data = np.append(data, train_data[i], axis=0)

# As n primeiras colunas representam os comprimentos de onda e a última
# coluna contém os labels para as regiões

data_x = data[:,:d]
data_y = data[:,-1]

# Normalização para manter os pixels entre 0 e 1

data_x = (data_x/255).astype(np.float16)

# Separação entre o conjunto de teste e de treinamento

X_train, X_test, y_train, y_test = train_test_split(data_x, data_y,
test_size=0.2, random_state=42)
```

A.5 Treinamento dos modelos

```
    names = ["K-Nearest Neighbors", "Linear SVM", "RBF SVM",
    "Random Forest"]

classifiers = [
    KNeighborsClassifier(n_neighbors=n_labels),
    LinearSVC(C=1),
    SVC(gamma=2, C=1),
    RandomForestClassifier(n_estimators=100)
    ]
```

```
models = []

for name, clf in zip(names, classifiers):

    print('Training ' + name  +' algorithm...')
    start_model = time.time()
    clf.fit(X_train, y_train)
    models.append(clf)
    end_model = time.time()
    print('Time spent:', end_model-start_model, '\n')
```

A.6 Teste dos algoritmos

```
for name, clf in zip(names, classifiers):

    print('Testing ' + name  +' algorithm...')
    score = clf.score(X_test, y_test)
    print('Accuracy:', score)
```

A.7 Classificação das imagens

```
for i, img in enumerate(test):
    w, h, d = img.shape
    img = img.reshape([w*h,d])
    img_predict = []

    for j, clf in enumerate(models):
        print('Predicting image', i, '-', 'model', names[j])
        img_predict.append(clf.predict(img))
        result = imfun.vector_to_image(img_predict[-1], n_labels, w, h)

        # Comandos para salvar a imagem resultante

        plt.figure()
        plt.imshow(result)
        plt.axis('off')
        filename = 'Image' + str(i) + '-' + 'model' + names[j]
```

```
        plt.savefig(filename, format='png')
        plt.close()
```

A função "vector_to_image" é a que realiza a coloração da imagem.

A.8 Filtro da mediana

```
result_mediana = []
result_mediana_final = []

for i, img in enumerate(img_predic):

    img = img.reshape([w,h]).astype(np.uint8)
    result_mediana.append(medianBlur(img, ksize=15))
    result_mediana_final.append(imfun.vector_to_image(
    result_mediana[-1].reshape([w*h]), n_labels, w, h))
```

A.9 Cálculo da acurácia final

```
# Define para qual imagem está sendo calculada a
# acurácia (1 ou 2)

imagem = 1

result1 = [] # Sem mediana
result2 = [] # Com mediana

test_data1 = [] # Sem mediana
test_data2 = [] # Com mediana

# Loop para criar uma lista dos resultados dos quatro modelos
#para uma imagem

for i in range(4):
    result1.append(img_predict.reshape([w,h]))
    result2.append(result_mediana.reshape([w,h]))

# Cria imagem RGB para usar na seleção das regiões
```

```
rgb_temp, bgr_temp, _ = imfun.create_rgb((test[imagem-1]*255).astype
(np.uint8), wavelength)

# Loop para selecionar manualmente as regiões utilizadas no cálculo
#da acurácia

for i in range(len(labels)):
    print('Select the region of', labels[i])
    data_temp1, data_temp2 = imfun.accuracy_result(result1, result2,
    bgr_temp, j)
    test_data1.append(data_temp1)
    test_data2.append(data_temp2)

test_data_final1 = []
test_data_final2 = []

# Loop para juntar as regiões em uma única lista

for j in range(len(test_data1)):
    if type(test_data1[j]) != int:
        for k in range(len(test_data1[j])):
            test_data_final1.append(test_data1[j][k])
            test_data_final2.append(test_data2[j][k])

accuracy1, accuracy2 = imfun.calculate_accuracy(test_data_final1,
test_data_final2)

# "accuracy1" se refere às imagens sem mediana e "accuracy2"
#se refere às imagens com mediana.
```

A.10 Algumas funções da biblioteca imfun

```
# Função para carregar as imagens multiespectrais

def open_images(folder):

    os.chdir(folder+'\\Imagens treinamento')
```

```
    folders = os.listdir()
    train_images = []
    for f in folders:
        train_images.append(load_multi_image(f))
        os.chdir('..')

    os.chdir(folder+'\\Imagens teste')
    folders = os.listdir()
    test_images = []
    for f in folders:
        test_images.append(load_multi_image(f))
        os.chdir('..')

    return train_images, test_images

# Função para criar a imagem RGB

def create_rgb(img, wavelength, red_wl=470, green_wl=570, blue_wl=630):

    blue = np.where(wavelength == blue_wl)[0][0]
    green = np.where(wavelength == green_wl)[0][0]
    red = np.where(wavelength == red_wl)[0][0]

    bgr = np.uint8(np.dstack((img[:,:,blue],img[:,:,green],
    img[:,:,red])))
    rgb = np.uint8(np.dstack((img[:,:,red],img[:,:,green],
    img[:,:,blue])))

    rgb_index = (red, green, blue)

    return rgb, bgr, rgb_index

# Função para normalizar as imagens

def normalize_glass(img, bgr, wavelength):

    norm_img = np.zeros(img.shape)
    print('Select a rectangle containing a small glass region')
    _, points = crop_image(bgr, show=False, cmap=None)
```

```
    for i in range(img.shape[2]):
        norm_img[:,:,i] = img[:,:,i]/np.mean(
        img[points[0][1]:points[1][1], points[0][0]:points[1][0],i])

    # Normalização dos valores entre 0 e 1

    for i in range(img.shape[2]):
        norm_img[:,:,i] = norm_img[:,:,i]*255/np.max(norm_img[:,:,i])

    norm_img = norm_img.astype(np.uint8)

    rgb, bgr, _ = create_rgb(norm_img, wavelength)

    return norm_img, rgb, bgr

# A função "crop_image" é uma simples função para cortar a imagem


# Função para selecionar manualmente as regiões para o treinamento

def define_labels(img, bgr, label):

    data = []
    keep = 'y'

    while keep == 'y':
        poly, points = polyroi(bgr)
        if len(points) > 2:
            poly = poly[:,:,0]
            temp = np.asarray(np.where(poly==255)).T
            for i in range(temp.shape[0]):
                data.append(np.append(img[temp[i,0],temp[i,1],:], label))
            keep = input('Do you wish to select more of this region?
            [y/n]: ')
        else:
            data = 0
            keep = 'n'

    return data
```

```
# A função polyroi é uma simples função para selecionar uma região
# poligonal da imagem

# Função para selecionar manualmente as regiões para calcular
# a acurácia final

def accuracy_result(result1, result2, bgr, label):

    data1 = []
    data3 = []
    keep = 'y'

    while keep == 'y':
        poly, points = polyroi(bgr)
        if len(points) > 2:
            poly = poly[:,:,0]
            temp = np.asarray(np.where(poly==255)).T
            for i in range(temp.shape[0]):

                data1.append([result1[0][temp[i,0],temp[i,1]],
                result1[1][temp[i,0],temp[i,1]],
                result1[2][temp[i,0],temp[i,1]],
                result1[3][temp[i,0],temp[i,1]],
                label])

                data2.append([result2[0][temp[i,0],
                temp[i,1]], result2[1][temp[i,0],
                temp[i,1]], result2[2][temp[i,0],
                temp[i,1]], result2[3][temp[i,0],
                temp[i,1]],
                label])

            keep = input('Do you wish to select more of this region?
            [y/n]: ')
        else:
            data1 = 0
            data2 = 0
            keep = 'n'
```

```
    return data1, data2

# Função para calcular a acurácia final

def calculate_accuracy(test_data1, test_data2):

    n = len(test_data1)

    acc1 = []
    acc2 = []

    for i in range(4):
        count1 = 0
        count2 = 0
        for j in range(n):
            if test_data1[j][i] == test_data1[j][-1]:
                count1 += 1
            if test_data3[j][i] == test_data2[j][-1]:
                count2 += 1
        acc1.append(count1/n)
        acc2.append(count2/n)

    return acc1, acc2

# Função para colorir a imagem e formatá-la para as dimensões originais

def vector_to_image(img, n_colors, w, h):

    colors = np.array([[200,200,0],[128,0,0],[0,128,0],[0,0,128],
    [128,128,128],[0,128,128],[128,0,128],[0,0,0],[255,255,255],
    [0,200,0],[200,0,0]])

    # 1 - yellow, 2 - red, 3 - green, 4 - blue, 5 - gray, 6 - cyan,
    7 - magenta, 8 - black, 9 - white

    codebook = colors[:n_colors,:]
    index = 0
    image = np.zeros([w,h,3])
```

```
for x in range(w):
    for y in range(h):
        image[x,y] = codebook[int(img[index]),:]
        index += 1
return np.uint8(image)
```